assignment 4.2.1

1:

¬¬p→p

1. F(¬¬p→p) ✓
2. T(¬¬p) 1,F→ ✓
3. F(¬p) 2,T¬ ✓
4. T(p) 3,F¬ ✓
5. F(p) 1,F→ ✓

P can not be both True and False, therefore the model is valid

2:

```
           ((p→q)→p)→p
           1. F(((p→q)→p)→p) ✓
           2. T((p→q)→p) 1,F→ ✓
           3. F(p) 1,F→ ✓
           /                      \
4. F(p→q) 2,T→ ✓            5. T(p) 2,T→ ✓
                            6. T(p) 4,F→ ✓
                            7. F(q) 4,F→ ✓
```

Both branches require P to be both True and False, therefore the model is valid

3:

```
           (¬p→¬q)→(q→p)
           1. F((¬p→¬q)→(q→p)) ✓
           2. F(q→p) 1,F→ ✓
           3. T(q) 2,F→ ✓
           4. F(p) 2,F→ ✓
           5. T(¬p→¬q) 1,F→ ✓
           /                \
6. F(¬p) 5,T→ ✓        7. T(¬q) 5,T→ ✓
8. T(p) 6,F¬ ✓         8. F(q) 7,T¬ ✓
```

The left branch requires P to be both True and False and is therefore closed. The right branch requires Q to be both True and False and is also closed. The model is valid

4:

```
           p→(p∧(q∨p))
           1. F(p→(p∧(q∨p))) ✓
           2. T(p) 1,F→ ✓
           3. F(p∧(q∨p)) 1,F→ ✓
           /                \
4. F(p) 3,F∧ ✓         5. F(q∨p) 3,F∧ ✓
                       6. F(q) 5,F∨ ✓
                       7. F(P) 5,F∨ ✓
```

Both branches require P to be both True and False, therefore the model is valid

5:

```
                              (pV(q∧r))→((pVq)∧(pVr))
                              1. F((pV(q∧r))→((pVq)∧(pVr))) ✓
                              2. T(pV(q∧r)) 1,F→ ✓
                              /                \
                  3. T(p) 2,TV ✓          4. T(q∧r) 2,TV ✓
                  /         \             5. T(q) 4,T∧ ✓
5. F(pVq) 7,F∧ ✓       6. F(pVr) 7,F∧ ✓  6. T(r) 4,T∧ ✓
7. F(p) 8,FV ✓         8. F(p) 9,FV ✓    7. F((pVq)∧(pVr)) 1,F→ ✓
9. F(q) 8,FV ✓         10. F(r) 9,FV ✓         /       \
                                              /         \
                                  8. F(pVq) 7,F∧ ✓      9. F(pVr) 7,F∧ ✓
                                  10. F(p) 8,FV ✓       11. F(p) 9,FV ✓
                                  12. F(q) 8,FV ✓       13. F(r) 9,FV ✓
```

Both left branches require P to be both True and False, and are therefore closed. The rigt branches require respectively Q and R to be False and True and are therefore also closed. The model is valid.

assignment 4.4.1

1:
¬¬p→p
Negate entire model
1. ¬(¬¬p→p)
NNF
2. ¬(p→p)
3. p∧¬p
CNF
4. [[p],[¬p]]

2:
((p→q)→p)→p
Negate entire model
1. ¬(((p→q)→p)→p)
NNF
2. ((p→q)→p)∧¬p
3. (¬(p→q)Vp)∧¬p
4. (p∧¬qVp)∧¬p
CNF
5. ((pVp)∧(¬qVp))∧¬p
6. [[p,p],[¬q,p],[¬p]]

3:
(¬p→¬q)→(q→p)
Negate entire model
1. ¬((¬p→¬q)→(q→p))
NNF
2. (¬p→¬q)∧¬(q→p)
3. (¬p→¬q)∧(q∧¬p)
4. (¬¬pV¬q)∧(q∧¬p)
5. (pV¬q)∧(q∧¬p)
CNF list of lists
6. [[p,¬q],[q],[¬p]]

4:
p→(p∧(q∨p))
Negate entire model
1. ¬(p→(p∧(q∨p)))
NNF
2. p∧(¬(p∧(q∨p)))
3. p∧(¬p∨¬(q∨p))
4. p∧(¬p∨(¬q∧¬p))
CNF list of lists
5. p∧((¬p∨¬q)∧(¬p∨¬p))
6. [[p],[¬p,¬q],[¬p,¬p]]

5:
(p∨(q∧r))→((p∨q)∧(p∨r))
Negate entire model
1. ¬((p∨(q∧r))→((p∨q)∧(p∨r)))
NNF
2. (p∨(q∧r))∧(¬((p∨q)∧(p∨r)))
3. (p∨(q∧r))∧(¬(p∨q)∨¬(p∨r))
4. (p∨(q∧r))∧((¬p∧¬q)∨(¬p∧¬r))
CNF list of lists
5. ((p∨q)∧(p∨r))∧((¬p∧¬q)∨(¬p∧¬r))
6. ((p∨q)∧(p∨r))∧((¬p∨(¬p∧¬r))∧(¬q∨(¬p∧¬r)))
7. ((p∨q)∧(p∨r))∧((¬p∨¬p)∧(¬p∨¬r)∧(¬q∨¬p)∧(¬q∨¬r))
8. [[p,q],[p,r],[¬p,¬p],[¬p,¬r],[¬q,¬p],[¬q,¬r]]

assignment 4.4.5 (answers from 4.4.1 as input)
1:
[[p],[¬p]]
[[],[]]
2 empty clauses

2:
¬p∨q∧¬p∨p
[[p,p],[¬q,p],[¬p]]
[[p,p],[¬q],[]]
1 empty clause

3:
[[p,¬q],[q],[¬p]]
[[p],[],[¬p]]
[[],[],[]]
3 empty clauses

4:
[[p],[¬p,¬q],[¬p,¬p]]
[[],[¬q],[¬p,¬p]]
1 empty clause

5:
[[p,q],[p,r],[¬p,¬p],[¬p,¬r],[¬q,¬p],[¬q,¬r]]
[[q],[p,r],[¬p],[¬p,¬r],[¬q,¬p],[¬q,¬r]]
[[],[p,r],[¬p],[¬p,¬r],[¬p],[¬q,¬r]]
[[],[],[¬p],[],[¬p],[¬q,¬r]]
3 empty clauses

First order logic axioms regarding spatial relations:

- part_of is transitive

$\forall x \forall y \forall z(\text{part\_of}(x,y) \rightarrow (\text{part\_of}(y,z) \rightarrow \text{part\_of}(x,z)))$

- if there is support, there is touch.

$\forall x \forall y(\text{support}(x,y) \rightarrow \text{touch}(x,y))$

- if there is touch, there is no near.

$\forall x \forall y(\text{touch}(x,y) \rightarrow \neg \text{near}(x,y))$

- therefore, if there is support, there is no near.

$\forall x \forall y(\forall \text{support}(x,y) \rightarrow \neg \text{near}(x,y))$

- touch is symmetric.

$\forall x \forall y(\text{touch}(x,y) \rightarrow \text{touch}(y,x))$

- near is symmetric.

$\forall x \forall y(\text{near}(x,y) \rightarrow \text{near}(y,x))$

- if there is part_of, there is no support and no touch.

$\forall x \forall y(\text{part\_of}(x,y) \rightarrow (\neg \text{support}(x,y) \wedge \neg \text{touch}(x,y)))$

- if there is support, near is for both.

$\forall x \forall y \forall z(\text{support}(x,y) \rightarrow (\text{near}(x,z) \wedge \text{near}(y,z)))$

- if there is part-of, near is for both.

$\forall x \forall y \forall z(\text{part\_of}(x,y) \rightarrow (\text{near}(x,z) \wedge \text{near}(y,z)))$

something can't be part of itself

$\forall x \neg \text{part\_of}(x,x)$